Corrigé DM3

Partie 1

1. Quel que soit L, $\varepsilon^{-1}L = L$. En effet, si $m \in \Sigma^\star$, $m \in \varepsilon^{-1}L \Leftrightarrow \varepsilon m = m \in L$.

2. Soit L un langage et $u, v \in \Sigma^\star$. Soit $m \in \Sigma^\star$. Alors

$$\begin{aligned} m \in u^{-1}(v^{-1}L) &\Leftrightarrow um \in v^{-1}L \\ &\Leftrightarrow vum \in L \\ &\Leftrightarrow m \in (vu)^{-1}L \end{aligned}$$

   On en déduit sans autre forme de procès l'égalité des langages $u^{-1}(v^{-1}L)$ et $(vu)^{-1}L$.

3. a) Le langage L_{ab} est l'ensemble des mots contenant ab. Les deux mots proposés contiennent déjà le motif ab, on peut donc les compléter par n'importe quoi et rester dans L_{ab}. On en déduit que $(bab)^{-1}L = (aaaba)^{-1}L = (a+b)^\star$.

   b) On a en fait $u^{-1}L_{ab} = \begin{cases} (a+b)^\star \text{ si } u \in L_{ab} \\ L + b(a+b)^\star \text{ si } u \notin L_{ab} \text{ et finit par un } a. \\ L \text{ si } u \notin L_{ab} \text{ et finit par un } b \end{cases}$

   Les trois conditions ci-dessus couvrent exhaustivement tous les cas et le langage L_{ab} a ainsi trois résiduels.

4. a) Soit $q \in Q$. Comme A est accessible, il existe un mot u permettant d'atteindre l'état q depuis q_0. Par ailleurs, le déterminisme de A assure que $\delta^\star(q_0, u)$ est bien réduit à un seul état : q. En particulier, il existe bien un mot u tel que $\delta^\star(q_0, u) = q$ ce qui permet de construire l'image de q par φ.

   Il reste à montrer que cette image ne dépend pas du choix du mot u. Pour ce faire, considérons deux mots u et v tels que $\delta^\star(q_0, u) = \delta^\star(q_0, v)$. Montrons qu'alors $u^{-1}L = v^{-1}L$. Si $m \in \Sigma^\star$, on a :

$$\begin{aligned} m \in u^{-1}L &\Leftrightarrow um \in L \text{ par définition d'un résiduel} \\ &\Leftrightarrow \delta^\star(q_0, um) \in F \text{ puisque } A \text{ reconnaît } L \\ &\Leftrightarrow \delta^\star(\delta^\star(q_0, u), m) \in F \\ &\Leftrightarrow \delta^\star(\delta^\star(q_0, v), m) \in F \text{ par hypothèse sur } u \text{ et } v \\ &\Leftrightarrow \delta^\star(q_0, vm) \in F \\ &\Leftrightarrow vm \in L \Leftrightarrow m \in v^{-1}L \end{aligned}$$

   Ainsi le résiduel associé à l'état q par φ est indépendant du choix du mot u.

   b) Montrons que φ est surjective. Soit $u^{-1}L$ un résiduel de L. Comme A est complet, le mot u peut être lu à partir de q_0 et il existe donc un état $q \in Q$ tel que $\delta^\star(q_0, u) = q$. En particulier, $u^{-1}L = \varphi(q)$.

   Ceci garantit que $|Q| \geq |\{u^{-1}L \,|\, u \in \Sigma^\star\}|$. Mais l'automate A est un automate fini donc $|Q|$ est fini ce qui garantit que le langage L a un nombre fini de résiduels.

5. a) La bonne définition de la fonction δ est en fait garantie par la question 2. La question demande de montrer que lire la même lettre depuis un même état produit le même résultat ce qui est évident.

   b) Par hypothèse, l'ensemble des résiduels de L — qui est aussi l'ensemble des états de $M(L)$ — est fini donc $M(L)$ est un automate fini. Son caractère déterministe et complet est immédiat par construction de sa fonction de transition et le fait qu'il ait un seul état initial.

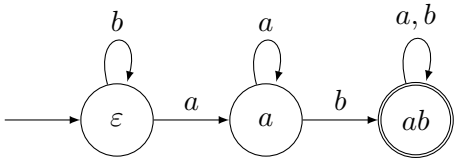
c) Procédons pour ce faire par double inclusion.

Si $m \in L$, $\delta^\star(\varepsilon^{-1}L, m) = m^{-1}L$ en lisant le mot m lettre à lettre et en utilisant la question 2. Or $m^{-1}L \in F$ puisque $m \in L$. On en déduit que lire m depuis l'état initial de $M(L)$ conduit à un état final de $M(L)$, c'est-à-dire que m est reconnu par $M(L)$.

Réciproquement, si m est reconnu par $M(L)$ alors il existe $u \in L$ tel que $m^{-1}L = \delta^\star(\varepsilon^{-1}L, m) = u^{-1}L$. Comme $u \in L$, $\varepsilon \in u^{-1}L$ et donc on a aussi $\varepsilon \in m^{-1}L$ ce qui garantit que $m \in L$.

Les questions 4 et 5 montrent qu'un langage est reconnaissable par un automate fini si et seulement si il admet un nombre fini de résiduels. Les langages reconnaissables étant les langages rationnels d'après le théorème de Kleene, on a bien prouvé que les langages rationnels sont les langages ayant un nombre fini de résiduels.

6. On connaît les résiduels de L_{ab} d'après la question 3b. On obtient ainsi :



Remarque 1. La relation R sur $\Sigma^\star$ définie par : u et v sont en relation si et seulement si $u^{-1}L = v^{-1}L$ est clairement une relation d'équivalence. Pour chaque classe selon cette relation, on choisit un représentant simple pour caractériser le résiduel correspondant. Par exemple, a est choisi pour représenter le résiduel $L + b(a+b)^\star$ car a est un mot "simple" qui n'appartient pas à L et termine par a.

Remarque 2. Constatez que l'automate obtenu est exactement l'automate des occurrences pour le motif ab ! C'est normal : l'automate des occurrences d'un motif est toujours l'automate minimal (donc l'automate des résiduels, cf question 7) permettant de reconnaître ce motif.

7. Tout d'abord, l'automate $M(L)$ est déterministe, complet et reconnaît L d'après la question 5.

Sa minimalité... a déjà été montrée en question 4 ! En effet, soit A un automate déterministe et complet dont l'ensemble d'états est Q et reconnaissant L. On peut rendre A accessible en l'émondant ce qui produit un automate A' dont l'ensemble d'états Q' est plus petit que Q. Comme la fonction φ définie en question 4 est surjective lorsque l'automate auquel elle se rapporte est déterministe, complet et accessible (cf question 4b), on a :

$$\underbrace{|\{u^{-1}L \,|\, u \in \Sigma^\star\}|}_{\text{nombre d'états de } M(L)} \leq |Q'| \leq |Q|$$

Donc, tout automate déterministe et complet reconnaissant L a plus d'états que $M(L)$ ce qui conclut.

Partie 2

8. Nulle et non avenue.

9. a) L'ensemble $Q/\sim$ est tout autant fini que Q donc $A/\sim$ a un nombre fini d'états. Pour montrer sa bonne définition il ne reste plus qu'à montrer que $\overline{\delta}(\overline{q}, a)$ ne dépend pas du représentant q choisi. Mais c'est immédiat d'après la propriété (2).

b) On montre tout d'abord par récurrence immédiate et à l'aide de la définition de $\overline{\delta}$ que, pour tout $m \in \Sigma^\star$, $\overline{\delta}^\star(\overline{q_0}, m) = \overline{\delta^\star(q_0, m)}$. On procède ensuite directement par équivalences :

$$
\begin{aligned}
m \in L(A) &\Leftrightarrow \delta^\star(q_0, m) \in F \\
&\Leftrightarrow \forall q \in \overline{\delta^\star(q_0, m)}, q \in F \text{ d'après la propriété (1) d'une congruence} \\
&\Leftrightarrow \forall q \in \overline{\delta}^\star(\overline{q_0}, m), q \in F \\
&\Leftrightarrow \overline{\delta}^\star(\overline{q_0}, m) \in \overline{F} \text{ d'après la définition de } \overline{F} \\
&\Leftrightarrow m \in L(A/\!\sim)
\end{aligned}
$$

10. La relation $\sim$ est immédiatement réflexive, symétrique et transitive : c'est une relation d'équivalence.

    Soit $q, q' \in Q$ tel que $q \sim q'$. Si q est final $\delta^\star(q, \varepsilon) = q$ est final donc $\varepsilon \in L_q = L'_q$ et ceci montre que $\delta^\star(q', \varepsilon) = q'$ est final. Le raisonnement réciproque tient également et donc la propriété (1) est vérifiée.

    Si $a \in \Sigma$ et $m \in \Sigma^\star$ tel que $m \in L_{\delta(q,a)}$ alors $\delta^\star(\delta(q, a), m)$ est final donc $\delta^\star(q, am)$ l'est donc $\delta^\star(q', am)$ aussi puisque $q \sim q'$. On en déduit donc que $m \in L_{\delta(q',a)}$. Cette inclusion est en fait une égalité par un raisonnement similaire et ceci montre que $\delta(q, a) \sim \delta(q', a)$ c'est-à-dire la propriété (2).

11. L'automate $A/\!\sim$ est bien défini et reconnaît $L(A) = L$ d'après la question 9. Il est déterministe et complet vu sa fonction de transition. Il ne reste plus qu'à montrer qu'il est minimal et par unicité de l'automate minimal, on concluera.

    Considérons la fonction suivante :

    $$
    \psi : \begin{cases} \{\overline{q} \,|\, q \in Q\} & \longrightarrow \{u^{-1}L \,|\, u \in \Sigma^\star\} \\ \overline{q} & \longmapsto L_q \end{cases}
    $$

    Elle est bien définie car pour tout $q \in Q$, L_q est effectivement un résiduel : c'est $m^{-1}L$ où m est l'un des mots (qui existe par accessibilité) permettant d'atteindre l'état q à partir de l'état initial dans A. De plus, si q, q' sont dans la même classe d'équivalence, par définition $L_q = L_{q'}$ donc $\varphi(\overline{q})$ ne dépend que de la classe $\overline{q}$ et non de son représentant q.

    D'autre part, par définition de $\sim$, si $L_q = L_{q'}$ alors $q \sim q'$ donc q et q' sont dans la même classe d'équivalence pour $\sim$. Cela montre l'injectivité de la fonction ψ. Il y a donc moins d'états dans $A/\!\sim$ que de résiduels de L c'est-à-dire que d'états dans l'automate des résiduels associé à L qu'on sait minimal parmi tous les automates reconnaissant $L = L(A)$ d'après la question 7. Ainsi, $A/\!\sim$ est bien minimal.

12. Le seul mot de longueur 0 est ε et $\delta^\star(q, \varepsilon) = q$ donc $q \sim_0 q'$ si et seulement si q et q' sont tous les deux finaux ou sont tous les deux non finaux. Il y a ainsi deux classes d'équivalence selon $\sim_0$: l'une contient tous les états finaux et l'autre tous les non finaux.

13. Soit $q, q' \in Q$ et $k \in \mathbb{N}$. Si $q \sim_{k+1} q'$ alors on a immédiatemment $q \sim_k q'$. De plus, si $a \in \Sigma$ et $m \in \Sigma^\star$:

    $$
    \begin{aligned}
    m \in \{u \in L_{\delta(q,a)} \,|\, |u| \leq k\} &\Leftrightarrow |m| \leq k \text{ et } m \in L_{\delta(q,a)} \\
    &\Leftrightarrow |m| \leq k \text{ et } \delta^\star(\delta(q, a), m) \in F \\
    &\Leftrightarrow |m| \leq k \text{ et } \delta^\star(q, am) \in F \\
    &\Leftrightarrow |m| \leq k \text{ et } \delta^\star(q', am) \in F \text{ car } q \sim_{k+1} q' \text{ et } |am| \leq k+1 \\
    &\Leftrightarrow m \in \{u \in L_{\delta(q',a)} \,|\, |u| \leq k\}
    \end{aligned}
    $$

    On en déduit que les langages $\{u \in L_{\delta(q,a)} \,|\, |u| \leq k\}$ et $\{u \in L_{\delta(q',a)} \,|\, |u| \leq k\}$ sont égaux et c'est la définition de $\delta(q, a) \sim_k \delta(q', a)$.

Réciproquement, soit m de longueur inférieure à $k+1$. Si $|m| \leq k$, comme $q \sim_k q'$ par hypothèse, $m \in L_q \Leftrightarrow m \in L_{q'}$. Sinon, il existe $a \in \Sigma$ et v de longueur k tel que $m = av$. Alors :

$$\begin{aligned} m \in L_q &\Leftrightarrow \delta^\star(q, m) \in F \\ &\Leftrightarrow \delta^\star(\delta(q,a), v) \in F \\ &\Leftrightarrow \delta^\star(\delta(q',a), v) \in F \text{ car } \delta(q,a) \sim_k \delta(q',a) \text{ et } |v| \leq k \\ &\Leftrightarrow \delta^\star(q', m) \in F \\ &\Leftrightarrow m \in L_{q'} \end{aligned}$$

On en déduit l'égalité des langages $\{u \in L_q \,|\, |u| \leq k+1\}$ et $\{u \in L_{q'} \,|\, |u| \leq k+1\}$ et donc $q \sim_{k+1} q'$.

14. Montrons par récurrence sur $l \in \mathbb{N}^\star$ que $\sim_k$ et $\sim_{k+l}$ sont égales. Pour $l = 1$, c'est immédiat vu l'hypothèse de la question. Soit maintenant $l \in \mathbb{N}^\star$ tel que $\sim_k$ et $\sim_{k+l}$ soient égales. On souhaite montrer que $\sim_k$ et $\sim_{k+l+1}$ le sont aussi. Or,

$$\begin{aligned} q \sim_{k+l+1} q' &\Leftrightarrow q \sim_{k+l} q' \text{ et } \forall a \in \Sigma, \delta(q,a) \sim_{k+l} \delta(q',a) \text{ par la question 13} \\ &\Leftrightarrow q \sim_k q' \text{ et } \forall a \in \Sigma, \delta(q,a) \sim_k \delta(q',a) \text{ par hypothèse de récurrence} \\ &\Leftrightarrow q \sim_{k+1} q' \text{ toujours d'après la question 13} \\ &\Leftrightarrow q \sim_k q' \text{ puisque } \sim_k \text{ et } \sim_{k+1} \text{ coïncident} \end{aligned}$$

15. On sait que $\sim_k$ et $\sim_{k+1}$ coïncident. La question 14 montre alors que pour tout $l \geq 0$, $\sim_k$ et $\sim_{k+l}$ coïncident. Ainsi, si $q \sim_k q'$ on a pour tout $l \geq 0$ l'égalité des deux langages suivants :

$$\{u \in L_q \,|\, |u| \leq k+l\} = \{u \in L_{q'} \,|\, |u| \leq k+l\}$$

En faisant l'union sur l de ces langages on obtient alors

$$L_q = \bigcup_{n \geq 0} \{u \in L_q \,|\, |u| \leq n\} = \bigcup_{l \geq 0} \{u \in L_q \,|\, |u| \leq k+l\} = \bigcup_{l \geq 0} \{u \in L_{q'} \,|\, |u| \leq k+l\} = L_{q'}$$

Comme $L_q = L'_q$, on a $q \sim q'$. Réciproquement, si $q \sim q'$ alors $q \sim_k q'$ s'ensuit immédiatemment. On en déduit que $\sim$ et $\sim_k$ décrivent les mêmes classes d'équivalence ce qui conclut.

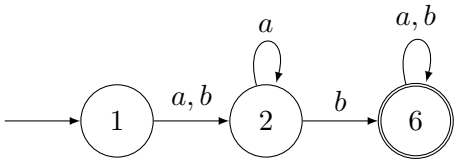
16. La question 13 explique que les classes d'équivalence pour $\sim_{k+1}$ se construisent en subdivisant éventuellement les classes d'équivalence pour $\sim_k$. Ainsi, si $\sim_k$ et $\sim_{k+1}$ diffèrent, alors le nombre de classes d'équivalence selon $\sim_{k+1}$ est strictement supérieur au nombre de classes selon $\sim_k$. La question 12 nous indique qu'initialement ce nombre de classes est 2 et comme ces classes d'équivalence partitionnent un ensemble de cardinal $|Q|$, leur nombre ne peut pas croître strictement au delà de $|Q|$.

Ainsi, au pire cas, $\sim_0$ compte 2 classes, $\sim_1$ en compte 3... $\sim_{|Q|-2}$ en compte $|Q|$ et $\sim_{|Q|-1}$ en compte $|Q|$ également. La question 15 montre alors que $\sim_{|Q|-2}$ correspond à $\sim$.

17. On calcule successivement les classes d'équivalence d'états de A selon $\sim_k$ grâce aux questions 12 et 13 :

k	classes d'équivalence selon $\sim_k$
0	$\{1,2,3,4,5\}$ $\{6\}$
1	$\{1\}$ $\{2,3,4,5\}$ $\{6\}$
2	$\{1\}$ $\{2,3,4,5\}$ $\{6\}$

La stabilisation est atteinte : les classes pour la congruence de Nerode sont données par la dernière ligne du tableau. On quotiente A selon le procédé décrit après la question 8, ce qui le minimise d'après la question 11. On obtient :



D'où on déduit immédiatemment que $(a+b)a^\star b(a+b)^\star$ est une expression rationnelle dénotant le langage reconnu par A.

Partie 3

18. a) Ce tableau est correct et c'est d'ailleurs le seul qui respecte $(\star)$.

    b) Ce tableau viole la deuxième condition de $(\star)$.

    c) Ce tableau ne représente même par les classes d'équivalence proposées puisqu'il impose que 3 et 4 soient dans la même classe, ce qui n'est pas.

19. La relation $\sim_0$ distingue deux classes d'après la question 12 : celle des états finaux et celle des non finaux. Pour respecter la deuxième condition de $(\star)$ il suffit de déterminer si l'état 0 est final ou non : s'il l'est, tous les finaux seront dans la classe 0, sinon, tous les finaux seront dans la classe 1.

```
int* initialiser_classes(afd *a)
{
    int n = a->taille_Q;
    int* classes = (int*)malloc(sizeof(int)*n);
    int numero_finaux;
    int numero_pas_finaux;
    if (a->finaux[0])
    {
        numero_finaux = 0;
        numero_pas_finaux = 1;
    }
    else
    {
        numero_pas_finaux = 0;
        numero_finaux = 1;
    }
    for (int i = 0; i < n ; i++)
    {
        if (a->finaux[i]) classes[i] = numero_finaux;
        else classes[i] = numero_pas_finaux;
    }
    return classes;
}
```

20. On lit dans `a->delta` l'état obtenu en lisant `l` depuis `q` puis on récupère la classe de ce dernier :

```
int classe_d_arrivee(afd* a, int* c, int q, int l)
{
    int etat_d_arrivee = a->delta[q][l];
    return c[etat_d_arrivee];
}
```

21. On s'aide de la question 13 : q et r sont dans la même classe selon $\sim_{k+1}$ s'il l'étaient déjà selon $\sim_k$ et si, pour toute lettre, la lecture de cette dernière dans q et r conduit à deux états dans la même classe selon $\sim_k$. Le résultat recherché est la négation de cette conjonction.

```
bool distinguables(afd* a, int* c, int q, int r)
{
    bool res = (c[q] == c[r]);
    for (int l = 0; l < a->taille_Sigma; l++)
    {
        res = res && (classe_d_arrivee(a,c,q,l) == classe_d_arrivee(a,c,r,l));
    }
    return (!res);
}
```

La complexité temporelle de distinguables est en $O(p)$ où p est le nombre de lettres de l'alphabet sur lequel est défini l'automate que cette fonction prend en entrée puisque classe_d_arrivee est en temps constant.

22. On propose le fonctionnement suivant. On utilise initialement une sentinelle -1 pour indiquer que les classes des états de a selon $\sim_{k+1}$ sont pour l'heure inconnues. Puis, on effectue l'opération suivante par ordre croissant de numéro d'état : si la classe de l'état considéré n'est pas encore connue, alors on lui assigne comme numéro de classe le plus petit numéro non encore utilisé, m, puis on détermine tous les états qui sont dans la même classe que lui. A la fin de ces instructions, tous les éléments de la classe m sont connus et on incrémente m avant de poursuivre.

```
int* raffiner(afd* a, int* c)
{
    int n = a->taille_Q;
    int* nouvelles_classes = (int*)malloc(sizeof(int)*n);
    for (int i = 0; i < n; i++)
    {
        nouvelles_classes[i] = -1;
    }
    int classe_courante = 0;
    for (int i=0; i<n; i++)
    {
        if (nouvelles_classes[i] == -1)
        {
            for (int j = 0; j < n; j++)
            {
                if (!distinguables(a,c,i,j)) nouvelles_classes[j] = classe_courante;
            }
            classe_courante++;
        }
    }
    return nouvelles_classes;
}
```

Si on note n le nombre d'états en entrée de raffiner et p le nombre de lettres de l'alphabet sur lequel il est défini, la complexité temporelle de raffiner est en $O(n^2p)$. En effet, l'initialisation de nouvelles_classes se fait en $O(n)$ puis, pour chacun des n états, on fait au plus n fois appel à la fonction distinguables.

Remarque : La boucle pour interne pourrait en fait commencer à i puisque la classe de tous les états qui lui sont inférieurs ont déjà été déterminées, par construction. Cela ne changerait pas l'expression de la complexité pire cas précédemment obtenue.

23. Comme remarqué en question 18a, les conditions ($\star$) imposent qu'il n'existe qu'un seul tableau permettant de représenter une partition donnée des états en classes d'équivalence. Le principe de nerode est ainsi de calculer les couples $\sim_k$ et $\sim_{k+1}$ pour k croissant et de vérifier si les deux tableaux correspondants sont égaux ou non. Si oui, c'est qu'on atteint stabilisation et la question 15 nous assure qu'on vient de trouver les classes selon la congruence de Nerode.

    On introduit donc une fonction tableaux_egaux permettant de tester si deux tableaux, supposés de même longueur, sont égaux :

```
bool tableaux_egaux(int* t1, int* t2, int n)
{
    bool res = true;
    for (int i = 0; i < n; i++)
    {
        res = res && (t1[i] == t2[i]);
    }
    return res;
}
```

    La fonction nerode est maintenant immédiate :

```
int* nerode(afd* a)
{
    int n = a->taille_Q;
    int* classes = initialiser_classes(a);
    int* classes_suivantes = raffiner(a,classes);

    while (!tableaux_egaux(classes,classes_suivantes,n))
    {
        int* temporaire = classes;
        classes = classes_suivantes;
        classes_suivantes = raffiner(a,classes_suivantes);
        free(temporaire);
    }
    free(classes_suivantes);
    return classes;
}
```

    En reprenant les notations de la fin de la question 22, on constate que nerode a une complexité temporelle en $O(n^3p)$. En effet, la question 16 assure que cette fonction effectuera au plus $n-2$ appels à raffiner, qui s'exécute quant à elle en $O(n^2p)$ d'après la question 15.

24. Le nombre d'états de l'automate minimal est le nombre de classes selon la congruence de Nerode. On calculera cette quantité (linéairement en la taille du tableau représentant les classes selon la congruence de Nerode, c'est-à-dire linéairement en le nombre d'états de l'automate à minimiser) via nombre_classes :

```
int nombre_classes(int* c, int n)
{
    int compteur = 0;
    for (int i = 0; i < n; i++)
    {
        if (c[i] == compteur)
        {
            compteur++;
        }
    }
    return compteur;
}
```

La fonction minimiser commence par calculer les classes selon la congruence de Nerode. La suite consiste à allouer correctement la mémoire nécessaire à la construction de l'automate minimal et à en remplir les champs à l'aide du procédé décrit juste après la question 8.

```
afd* minimiser(afd* a)
{
    int nb_etats = a->taille_Q;
    int* classes = nerode(a);
    int nb_classes = nombre_classes(classes, nb_etats);

    afd* min = (afd*)malloc(sizeof(afd));
    min->taille_Q = nb_classes;
    min->taille_Sigma = a->taille_Sigma;
    min->q0 = classes[a->q0];

    bool* classes_finales = (bool*)malloc(sizeof(int)*nb_classes);
    for (int i = 0; i < nb_etats; i++)
    {
        int classe = classes[i];
        classes_finales[classe] = a->finaux[i];
    }
    min->finaux = classes_finales;

    int** delta = (int**)malloc(sizeof(int*)*nb_classes);
    for (int i = 0; i < nb_classes; i++)
    {
        delta[i] = (int*)malloc(sizeof(int)*min->taille_Sigma);
    }
    for (int q = 0; q < nb_etats; q++)
    {
        int classe_depart = classes[q];
        for (int x = 0; x < min->taille_Sigma; x++)
        {
            int classe_arrivee = classe_d_arrivee(a,classes,q,x);
            delta[classe_depart][x] = classe_arrivee;
        }
    }
```

```
    min->delta = delta;

    free(classes);
    return min;
}
```

25. Le calcul de la congruence de Nerode se fait en $O(n^3p)$ d'après la question 23. L'initialisation des états finaux de l'automate minimal se fait linéairement en son nombre d'états, qui est majoré par n. De même, l'initialisation de ses transitions se fait en $O(np)$. Ainsi, la complexité de minimiser est en $O(n^3p)$.

Bonus

26. Soit A un automate reconnaissant L. L'automate $T(A)$ est déterministe donc n'a qu'un seul état initial qu'on note f. On en déduit que A a pour seul état final f et on peut donc écrire : $A = (\Sigma, Q', I', \{f\}, \eta)$. L'automate $D(A)$ est le morceau accessible de l'automate des parties associé à A donc $D(A) = (\Sigma, Q, \{i\}, F, \delta)$ avec $Q \subset P(Q')$ et $i = \{I'\}$. Considérons la fonction suivante (c'est celle de la question 4 !) :

$$\varphi : \begin{cases} Q \longrightarrow \mathcal{R}_L = \{u^{-1}L \,|\, u \in \Sigma^\star\} \\ q \longmapsto u^{-1}L \text{ où } u \text{ est un mot tel que } \delta^\star(I, u) = q \end{cases}$$

Comme $D(A)$ est déterministe et accessible, la question 4a montre que cette fonction est bien définie. Montrons qu'elle injective. Soit donc $q, q' \in Q$ tels que $\varphi(q) = \varphi(q')$. Alors il existe $u, v \in \Sigma^\star$ tels que $u^{-1}L = v^{-1}L$, $q = \delta^\star(I, u)$ et $q' = \delta^\star(I, v)$. On cherche à montrer que $\delta^\star(I, u) = \delta^\star(I, v)$ en tant qu'états de $D(A)$. Pour ce faire, on va montrer que ces états sont égaux en tant qu'ensembles d'éléments de Q'.

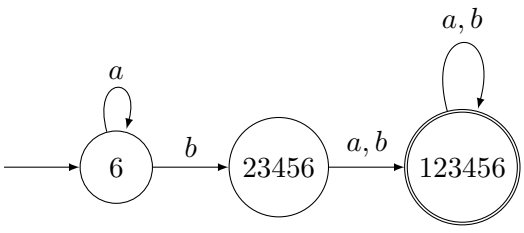
Soit $p \in Q'$ tel que $p \in \delta^\star(I, u)$. Alors $p \in \eta^\star(I', u)$. Comme $T(A)$ est accessible, il existe un chemin dans $T(A)$ depuis f vers p étiqueté par un certain mot. Donc il existe un chemin menant de p vers f dans A étiqueté par le miroir de ce mot, qu'on note h. Ainsi, $f \in \eta^\star(I', uh)$ donc uh est reconnu par A donc appartient à L. Donc $h \in u^{-1}L$ et par hypothèse $h \in v^{-1}L$, c'est-à-dire que $vh \in L$. Ainsi, $\eta^\star(\eta^\star(I', v), h)$ contient l'état final f de A. Cela signifie que lire h à partir d'un des états de $\eta^\star(I', v)$ mène à f dans A. Mais le fait que $T(A)$ soit déterministe impose qu'il n'y a qu'un seul chemin dans $T(A)$ étiqueté par h à partir de l'état f donc dans A il n'y a qu'un seul chemin permettant d'atteindre f en lisant h et on en connaît un : c'est celui qui part de p. On en déduit que $p \in \eta^\star(I', v)$ donc que $p \in \delta^\star(I, v)$.

L'autre inclusion se fait de la même façon et finalement $q = q'$. Comme φ est injective, $|Q| \leq |\mathcal{R}|$ donc $D(A)$ est un automate déterministe complet reconnaissant L qui a moins d'états que l'automate des résiduels qu'on sait minimal pour L : il est minimal aussi.

27. Si A reconnaît L, $T(A)$ reconnaît le langage miroir de L donc $D(T(A))$ aussi puis $T(D(T(A)))$ reconnaît le miroir du miroir de L c'est-à-dire L par involutivité de la transposition. Finalement, $D(T(D(T(A))))$ reconnaît L. Il est déterministe et complet d'après l'algorithme de déterminisation accessible.

Montrons à présent que cet automate est minimal. L'automate $B = T(D(T(A)))$ est un automate qui reconnaît L et dont le transposé, $T(B) = D(T(A))$ est déterministe et accessible puisque l'opérateur D déterminise accessiblement. La question 26 assure donc que $D(B) = D(T(D(T(A))))$ est minimal.

28. On transpose A ; en particulier, 1 devient final et 6 intial. Puis on déterminise pour obtenir :



On transpose cet automate, on le déterminise accessiblement et on obtient effectivement le même automate qu'à la question 17.